EXHIBIT 3
Part 3 of 4

63. Gorthy explains that the described XML configuration command (input command) is generated using an XML configuration schema. Ex. 1003 at 3:9-11; *see also* Clark Decl. ¶ 61. The XML configuration schema is generated "from the CLI commands associated with a Cisco™ router," and it "contains the commands, command hierarchy and bounds of the various configuration commands" of the router. Ex. 1003 at 3:11-13, 4:40-41; *see also id.* at 2:59-62; *see also* Clark Decl. ¶ 61. It is "in essence . . . a modeling of the router's command structure." *Id.* at 2:63-65, 3:9-11, 5:8-9. The XML configuration schema, and XML-based commands generated from that schema as described in Gorthy, thus have a "CLI syntax with CLI keywords sequenced according to configuration rules for CLI commands." Clark Decl. ¶ 61; *see also* Ex. 1003 at 3:13-16 ("When given a command in XML format, the command information in the configuration schema can be used to reformat the XML-based command into a proper CLI format.").

In Appendix B, Gorthy provides an example of an XML configuration schema corresponding to a portion of one of the CLI commands (the "service" command) in Appendix A. *See* Ex. 1003 at 5:31-34; *see also id.* at Appendix A and B. As reflected in Appendix B, the schema that controls whether a particular XML-based command will be accepted as valid requires that the particular command words be entered in the order in which they are found in valid CLI commands—in other words, that valid input commands must follow CLI syntax. Clark Decl. ¶ 62.

Consider, for example, the following:

```
<service>
  <alignment>
    <detection/>
  </alignment>
</service>
```

Appendix B would indicate to one of ordinary skill in the art that this XML-based command is a valid command because its XML tags are named consistent with, and according to the hierarchical relationships defined by, the schema. Clark Decl. ¶¶ 62-63; *see also* Ex. 1003 at Appendix B (The "<xsd:element name='detection'>" tag is nested within the "<xsd:element name='alignment'>" tag, which is nested within the "<xsd:element name ='service'>" tag.) Such an input command is thus in XML format having a CLI syntax with CLI keywords sequenced according to configuration rules for CLI commands. Clark Decl. ¶ 63.

4. [1C.1] "translating, with the CLI parser, the input command from the XML format having the CLI syntax into a CLI command that, when executed, is configured to prompt the routing system to perform the operation,"

Gorthy discloses translating, with the CLI parser, the input command from the XML format having the CLI syntax into a CLI command that, when executed, is configured to prompt the routing system to perform the operation. *Id.* at ¶¶ 64-65. As discussed above, Gorthy discloses translating XML-based configuration commands having CLI syntax to CLI-based commands using an XML schema. Ex. 1003 at 3:9-16 ("[T]he schema can be used to generate CLI commands from, for

example, XML-based commands. . . . When given a command in XML format, the command information in the configuration schema can be used to reformat the XML-based command into a proper CLI format."); 6:9-11 ("That XML-based command can be passed to the converter 235 which converts the XML-based command to a CLI-based command using the XML schema."); *see also* 6:35-40, 6:43-51. A POSA would understand that the XML "service" command above would be translated into the CLI command "service alignment detection." Clark Decl. ¶ 64; *see also* Ex. 1003 at Appendix B, Appendix A.

Gorthy also discloses that the converted CLI command, when executed, is configured to prompt the routing system to perform the operation responsive to the CLI command. Clark Decl. ¶ 65. Gorthy explains, for instance, that the converted CLI configuration command is transmitted through the network to the router where it is used to configure the router. Ex. 1003 at 3:16-19; *see also id.* at 6:9-14 ("Th[e] XML-based command can be passed to the converter 235 which converts the XML-based command to a CLI-based command using the XML schema. A CLI-based command, not the XML command, can then be passed to the configuration storage module 145 where it is integrated into the configuration of the router."); *id.* at 6:21-26 ("the localized converter 235′ and schema storage module 240′ convert the XML-based command to a CLI-based command and then transmit the CLI-

based command through the network 250 to the router 120"); *id*. at Fig. 8; *see also* claim element [1B]; Ex. 1003 at 4:33-35.

5. [1C.2] "wherein the translating of the input command into the CLI command includes identifying at least one XML tag that includes an XML parameter to indicate the XML tag includes one or more CLI keywords, extracting the one or more CLI keywords from the input command, and arranging the one or more CLI keywords into the CLI command according to the CLI syntax of the input command,"

Gorthy in view of Froyd discloses this limitation. Clark Decl. ¶¶ 66-74. As discussed above, Gorthy discloses translating an XML-based command to a CLI command using an XML configuration schema that models the command hierarchy of a given router. The exemplary XML configuration schema disclosed by Gorthy in Appendix B specifies a series of element name parameters (*e.g.*, "service," "alignment," "detection," "logging"). When viewed in comparison to Appendix A, which discloses exemplary CLI commands for a given router, a POSA would recognize that each of the element name parameters of Appendix B corresponds with a CLI keyword used in one or more CLI commands. Clark Decl. ¶ 67. Moreover, the element names in the XML schema are arranged in a hierarchical order that reflects the permissible sequencing of CLI keywords for a given CLI command. *Id.* For example, nested beneath the "<xsd:element name='service'>" tag is the "<xsd:element name='alignment'>" tag. *Id.*; *see also* Ex. 1003 at Appendix B. One level below that tag are the "<xsd:element name='detection'>" and

"<xsd:element name='logging'>" tags. Clark Decl. ¶ 67; *see also* Ex. 1003 at Appendix B. The initial CLI commands listed in Appendix A reflect this same hierarchical sequencing: "service," "service alignment," "service alignment detection," and "service alignment logging." *See* Ex. 1003 at Appendix A.

From the XML configuration schema of Appendix B, one of ordinary skill in the art would understand what a valid XML command according to that schema would look like. As discussed above, one example would be:

```
<service>
  <alignment>
    <detection/>
  </alignment>
</service>
```

Clark Decl. ¶ 68; *see also* discussion of claim 1B.2, above. As this example illustrates, this XML input command—which would translate to the example command "service alignment detection" in Appendix A—consists of a series of XML tags. Clark Decl. ¶ 68. Within each of the XML tags is a CLI keyword. *Id.* Because the XML configuration schema includes element name parameters *only* for CLI keywords, the CLI keywords in the XML tags are themselves parameters that indicate the tag includes a CLI keyword. *Id.*

A POSA would also recognize that alternative XML schemas could be used with the system disclosed by Gorthy, including schemas that permit XML tags to identify CLI keywords using separate XML parameters. *Id.* at ¶ 69. Froyd discloses

XML commands according to such a schema. For example, Froyd teaches that the following XML command could be used:

```
<command>
    <keyword text="show">
    </keyword>
</command>
```

Ex. 1004 at 8:58-61. As Froyd explains, "[t]he <keyword> tag 420 specifies the text of a command keyword in the command set of the CLI such as, for example, SHOW, ENABLE, or LOGOUT. The text is supplied by the single attribute 'text'." *Id.* at 8:63-66. Here, the XML tag <keyword . . .> includes an XML parameter ("text=") that indicates the tag includes a CLI keyword ("show"). Clark Decl. ¶ 69.

As the above examples of XML commands from Gorthy and Froyd illustrate, there are numerous ways to represent CLI keywords in XML. *Id.* at ¶ 70. One of ordinary skill in the art would understand that the decision of how to represent CLI keywords in XML is simply an implementation choice. *Id.* As these references illustrate, there was nothing novel in 2005—nearly four years after Froyd and Gorthy's applications were filed—about using a parameter to indicate that an XML tag includes a CLI keyword (as in Froyd), as opposed to an XML tag that contains only the CLI keyword (as in Gorthy). *Id.* A POSA would have recognized that using an XML parameter to indicate the presence of one or more CLI keywords was a non-inventive implementation decision, and would have found it obvious to use the

XML parameter disclosed in Froyd with the XML-to-CLI translation system of Gorthy. *Id.*

It also would have been obvious to a POSA that translating the XML command to a CLI command, as disclosed in Gorthy, could occur by identifying an XML tag that includes an XML parameter to indicate that the XML tag includes one or more CLI keywords. *Id.* at ¶ 72. As discussed above, the XML schema disclosed in Gorthy shows that the system is able to identify the CLI keywords in XML commands. Gorthy's schema identifies XML tags that include an XML parameter to indicate that the XML tag includes one or more CLI keywords because, *e.g.*, "service," "alignment," and "detection" are all valid element names as described and structured in the XML configuration schema. *Id.* at ¶ 73.

As discussed above, however, even if a CLI keyword needed to be identified by a separate XML parameter, Froyd illustrates that such an implementation was known in the art long before the '886 patent and would have been an obvious adaptation of the XML command disclosed by Gorthy. *Id.*

A POSA would likewise understand that the process of translating the input XML command into a CLI command involves extracting the one or more CLI keywords from the input command and arranging those CLI keywords into a CLI command according to the CLI syntax of the input command. *Id.* at ¶ 74. Gorthy's exemplary schema of Appendix B identifies CLI keywords that would be found in

XML tags of an input command in the order in which those CLI keywords would appear in a CLI command. *Id.* This is evidenced by Appendix A, which shows exemplary CLI commands that could be generated using an XML schema, such as that of Appendix B. *Id.* Note that the first three commands of Appendix A ("service," "service alignment," and "service alignment detection") include CLI keywords that are sequenced in the same order as those in the schema of Appendix B. *Id.* Viewing these examples, a POSA would find it obvious that in translating the input command in XML format (*e.g.*, "<service><alignment></detection></alignment></service>"), Gorthy extracts and arranges the CLI keywords into a CLI command according to the CLI syntax of the input command (*e.g.*, "service alignment detection"). *Id.*

6. [1C.3] "wherein the routing system is configured to perform the operation responsive to the execution of the CLI command;"

As discussed with reference to claim 1B.1 above, Gorthy discloses that the routing system is configured to perform the operation responsive to the execution of the CLI command. *Id.* at ¶ 75. Gorthy explains that "[o]nce reformatted into a CLI format, the command can be pushed out to the appropriate router. Thus, a system administrator could configure such a router without knowing the specifics of the CLI." Ex. 1003 at 3:16-19; *see also id.* at 4:33-35 ("[N]ew configuration commands are provided to the router 120 through the CLI."). Gorthy further explains

that a CLI-based command is "passed to the configuration storage module 145 *where it is integrated into the configuration of the router*." *Id.* at 6:12-15 (emphasis added). One of skill in the art would understand from this disclosure that the routing system is configured to perform the operation responsive to the execution of the CLI command. Clark Decl. ¶ 75. For example, when a "service alignment detection" CLI command (*see* Ex. 1003 at Appendix A) is passed to the router, one of skill in the art would understand that the router performs the operation responsive to execution of that CLI command (enabling detection of alignment issues). Clark Decl. ¶ 75.

7. [1D] "translating an output message, generated in response to performance of the operation, from a CLI format into an XML format having the CLI syntax, wherein the translating includes parsing the output message to identify at least one CLI token, translating each CLI token of the output message into a corresponding XML value according to a stored mapping of CLI tokens-to-XML values, and generating the output message in the XML format with the XML values;"

Gorthy in combination with Courtney, Froyd, and the JUNOScript Guide discloses this limitation. Clark Decl. ¶¶ 76-82. For instance, Courtney discloses a converter that translates an output message in a CLI format to an XML format. *See* Ex. 1002 at 2:40-45. A POSA would appreciate that one sort of output message that can be translated is an output message providing the current configuration of the system. Clark Decl. ¶ 76. This can be done through what Gorthy refers to as "a

command extraction mode," which "is activat[ed] by entering a '?' at the prompt." Ex. 1003 at 4:48-49. The submission of such a command leads to a response, or series of responses, that allow a user to "retrieve[] the primary commands, subcommands and bounds" of a system. *Id.* at 4:50-51; *see also* Ex. 1002 at 4:48-49 ("available commands are returned through the CLI"). Messages output in response to such commands requesting configuration information regarding a system can thereafter be converted into XML format: "In certain embodiments, this schema can be directly used to generate an XML document that represents the configuration of the particular network device." *Id.* at 3:12-14. When translating from CLI to XML, the obvious and most sensible approach to take would be to retain the same syntax as the output message being translated, thereby creating a message in an XML format with the CLI syntax. Clark Decl. ¶ 76. Moreover, the schema ensures that translated commands in XML format have proper CLI syntax. *Id.*

As part of the conversion from CLI to XML, Courtney's system analyzes the output message to extract at least one CLI token. *Id.* at ¶ 77. Courtney discloses that one of the first steps in a conversion is to "identif[y] each initial command within each configuration line" from the configuration of the device. Ex. 1002 at 7:17-21. Because such initial commands are CLI tokens, Courtney's system parses the file in order to extract at least one such token. Clark Decl. ¶ 77.

Courtney also discloses translating each CLI token of the output message into a corresponding XML value according to a stored mapping of CLI tokens-to-XML values. *Id.* at ¶ 78. "[U]sing the identified initial command, the XML converter 235 generates a look-up key that is used to index the hash table, locate a hash map object that corresponds to the look-up key and retrieve that hash map object (steps 275 and 280)." Ex. 1002 at 7:27-31. "The hash map object contains schema information regarding the command or value such as whether optional or required data type, etc." *Id.* at 7:31-33. "Finally, using this hash map object, the XML converter 235 can assemble the XML-based command and write it to the corresponding XML document (step 295)." *Id.* at 7:33-36. This process is repeated for each command in the network device's native-format configuration. *Id.* at 7:37-38. A POSA would also understand that the conversion from CLI tokens to XML values inherently requires that some mapping exist between the two and that they be stored, even if only in a transitory manner, to accomplish the desired conversion. Clark Decl. ¶ 78. The message will thereafter be converted into XML format, using the XML values. *Id.*

The JUNOScript Guide provides another example in the prior art of translating each CLI token of the output message into a corresponding XML value according to a stored mapping of CLI tokens-to-XML values. *Id.* at ¶ 79. The JUNOScript Guide explains that the output from a CLI command is, by default, for-

matted ASCII text. Ex. 1005 at 37. It provides the following example of an output message produced in response to a CLI command:

> Physical interface: **fxp0**, **Enabled**, Physical link is **Up**
> Interface index: **4**, SNMP ifIndex: **3**

Id. at 17 (emphasis added).

The JUNOScript Guide explains that the JUNOScript API translates this output message from a CLI format into an XML format having a CLI syntax. Clark Decl. ¶ 80. In doing so, JUNOScript parses the output message shown above to identify at least one CLI token and translates each token to a corresponding XML value, thus generating the output message in XML format with XML values:

```
<interface>
        <name>fxp0</name>
        <admin-status>enabled</admin-status>
        <operational-status>up</operational-status>
        <index>4</index>
        <snmp-index>3</snmp-index>
</interface>
```

Ex. 1005 at 17 (emphasis added); Clark Decl. ¶ 80.

In another example, the JUNOScript Guide explains that the JUNOScript API translates an output message, generated in response to the performance of an operation, from a CLI format into an XML format having a CLI syntax by using the "pipe" function and the "display xml" command:

```
user@host> show chassis hardware | display xml
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/5.1R1/dtd/junos.dtd">
<chassis-inventory xmlns="http://xml.juniper.net/junos/5.1R1/dtd/junos-chassis.dtd">
<chassis junos:style="inventory">
<name>Chassis</name>
<serial-number>00118</serial-number>
<description>M40</description>
<chassis-module>
<name>Backplane</name>
<version>REV 06</version>
<part-number>710-000073</part-number>
<serial-number>AA2049</serial-number>
</chassis-module>
<chassis-module>
<name>Power Supply A</name>
...
</chassis-module>
...
</chassis>
</chassis-inventory>
</rpc-reply>
```

Ex. 1005 at 37; Clark Decl. ¶ 81.

The JUNOScript Guide examples show that the output message generated in response to a CLI command is parsed to identify at least one CLI token (*e.g.*, "fxp0," "enabled," etc.), and that each CLI token is translated into a corresponding XML value (*e.g.*, "<name>fxp0</name>," "<admin-status>enabled</admin-status>"). Clark Decl. ¶ 82. The JUNOScript Guide recognizes that, by default, the output of a CLI command to the JUNOScript server will be in CLI format. Ex. 1005 at 37 ("To display the output from a JUNOS CLI command as JUNOScript tags rather than the default formatted ASCII, pipe the command to the display xml command."). By using the "| display xml" feature, the JUNOScript API translates the default CLI output to XML. *Id.* In doing so, it is both inherent and obvious that the JUNOScript API translates the output message "according to a stored mapping of CLI tokens-to-XML values," as there would be no other way for the software to produce the XML output from a pipe ("|") function, which uses the default CLI

command output as the input for the "display xml" function. Clark Decl. ¶ 82. In other words, because we know that a CLI format output message is the input to the "display xml" function, a POSA would understand that it would be obvious that the JUNOScript API accesses a stored mapping to produce the XML-format output message shown on page 37 of the JUNOScript Guide. *Id.* The result is an output message in XML format with the XML values, as shown in the above example. *Id.*

8. [1E] "and transmitting the output message in the XML format having the CLI syntax to a remote device external from the routing system."

Gorthy in combination with Courtney, Froyd, and the JUNOScript Guide discloses this limitation. *Id.* at ¶ 83. For instance, Courtney discloses that the output from the XML converter can be passed to relevant software applications, which a POSA would understand to include software applications running outside the routing system. *See* Ex. 1002 at 6:50-60; Clark Decl. ¶ 83. This is also confirmed by Courtney's statement that the configuration for a network device could be obtained from "an alternate location." *See* Ex. 1002 at 2:43. A POSA would understand that converted XML commands also could be transmitted to other types of remote devices and that one of the reasons to use an easily transferable language like XML is to facilitate such transfers. Clark Decl. ¶ 83. For instance, Froyd discloses presenting converted configuration data in XML format "to the user at the workstation." Ex. 1004 at 14:9-12; *see also id.* at 7:24-25 ("The data [is] then sent

to a destination (*e.g.*, user, management application, data file, etc."); *id.* at 15:60-63 ("Placing the statistic information into the XML format also allows the statistic information to be used by other devices capable of processing data in the XML format."). The JUNOScript Guide similarly teaches the transmitting the output message in XML format to a remote device external to the routing system, such as a client application. Ex. 1005 at 15 ("Client applications can configure or request information from a router by encoding the request with JUNOScript tags and sending it to the JUNOScript server running on the router. The JUNOScript server directs the request to the appropriate software modules within the router, encodes the response in JUNOScript tags, and returns the result to the client application."); Clark Decl. ¶ 83.

B. Claim 2

1. "The method of claim 1, wherein the input command is formatted in accordance with an XML schema of CLI rules and behaviors enforced by an internetwork operating system (IOS) command line interface (CLI) parser subsystem."

Gorthy in combination with Courtney, Froyd, and the JUNOScript Guide discloses this claim. *Id.* at ¶¶ 84-85. For instance, Gorthy discloses the use of Cisco routers to perform the claimed method. Ex. 1003 at 6:52-55 (explaining that "the embodiments of the present invention are generally described with regard to a router and in particular with regard to Cisco™ routers"). As explained above with respect to claim 1, Gorthy discloses using an XML configuration schema in con-

junction with a converter to generate CLI commands from XML-based commands. *Id.* at 2:63-3:11; Clark Decl. ¶ 61, 84. Gorthy teaches that this XML configuration schema can be generated from the CLI commands associated with the target router (including, for example, a Cisco router). Ex. 1003 at 4:39-41; Clark Decl. ¶ 84. "After the configuration schema has been generated, it is associated with characteristics of the router and stored accordingly." Ex. 1003 at 5:50-52. In particular, Gorthy explains that "the configuration schema might be associated with a Cisco™ router, model 7500, OS version 12.0." *Id.* at 5:52-54.

The term "internetwork operating system (IOS)," as recited in claim 2, is a term coined by Cisco that refers to operating system software that runs on Cisco routers. Clark Decl. ¶ 85. Therefore, a POSA would understand that the CLI rules and behaviors of the disclosed XML configuration schema, when designed to be used with Cisco routers, would be enforced by a component of the internetwork operating system (IOS) (*i.e.,* a command line interface (CLI) parser subsystem). *Id.*

C. Claim 3

1. [3A] "The method of claim 2, wherein the translation of the input command from XML format having a CLI syntax into a CLI command comprises:"

Gorthy in combination with Courtney, Froyd, and the JUNOScript Guide discloses the method of claim 3, as discussed below. Clark Decl. ¶ 86.

2. [3B] "parsing the input command to identify an XML command attribute;"

Gorthy in combination with Froyd discloses parsing the input command to identify an XML command attribute, for the reasons discussed above with respect to claim element 1C. *Id.* at ¶¶ 87-88. For instance, as discussed above, a POSA would understand that the process disclosed in Gorthy of translating the input command into a CLI command involves analyzing XML tags to identify keywords and parameters. *See* discussion of claim element 1C. Moreover, in Appendix B, Gorthy provides an example of an XML configuration schema that includes XML tags that contain XML parameters indicating that the tag contains one or more CLI keywords. Clark Decl. ¶ 87. The XML schema disclosed in Gorthy thus shows that the system is able to identify these things. *Id.*

Moreover, the Froyd patent, which specifically claims converting XML configuration files into CLI configuration commands, specifically discloses that an XML command in CLI syntax can be converted to a CLI command. Clark Decl. ¶ 88; s*ee* Ex. 1004 at 17:57-60 (claim 5); *see also id.* at 17:51-52 (claim 2). For instance, in the course of discussing Figure 4 (a table illustrating exemplary XML tags), Froyd explains that "[t]he <command> tag 415 is the top-level tag for a single command." Ex. 1004 at 8:51-52. Froyd further explains that "[t]he <command> tag 415 contains a single <keyword> tag" and that "[t]he <keyword> tag 420 specifies the text of a command keyword in the command set of the CLI such